AWT

Técnicas de Programación
Curso 2005/06

Santiago Pavón – UPM
V:060.423

¿Qué es **java.awt**?

- Contiene clases para crear GUIs.
- Historia
- Los interfaces de usuario se crean empleando clases que representan componentes:
  - botones, ventanas, etc.
- Existen componentes de tipo contenedor.
  - Contienen a otros componentes
  - Usan gestores de geometría
- Los componentes producen eventos
  - provocan la ejecución de ciertos métodos en las clases escuchadores.

Pasos para crear un GUI

1. El área donde crearemos el UI será el proporcionado por un contenedor, y que inicialmente será una ventana.
2. Definir el gestor de geometría que usaremos para el contenedor. Decidirá como se visualizan los componentes dentro del contenedor.
3. Crear los componentes que irán dentro del contenedor.
4. Añadir (visualizar) los componentes dentro del contenedor.
5. Enlazar con los escuchadores de eventos.
6. Si los componentes que creamos son contenedores se repite el ciclo otra vez.

- Derivar para cambiar comportamiento por defecto.

Métodos para repintar

- Los componentes muestran un contenido gráfico que es necesario repintar en determinadas ocasiones.
- **paint(Graphics g)**
  - Contiene el código para pintar.
- **update(Graphics g)**
  - Borra el componente y lo repinta (llama a **paint**).
  - Puede redefinirse para invocar sólo a paint, sin borrar previamente, evitando así parpadeos.
- **repaint()**
  - Repintar el componente.
  - Es el método normal a invocar por los demás métodos de la clase.
  - Llama a **update**.
- Son métodos heredados de **Component**.

Repintado automático

- La mayoría de los componentes se repintan solos automáticamente.

- Lo que no se repinte solo, lo debemos repintar escribiendo el código necesario dentro del método **paint** del componente.
  - por ejemplo: decoración incluida por nosotros

Ejemplo creación GUI



Ejemplo creación GUI

```
public static void main(String[]a) {
    Frame f = new Frame("Ejemplo");
    Main p = new Main();
    f.add(p);
  f.pack();
  f.setVisible(true);
}
```

Técnicas de Programación

Ejemplo creación GUI

```
public class Main extends Panel {

    public Main() {
      ?????
    }

    /** Pinta el borde rojo. */
    public void paint(Graphics g) {
      ??????
    }
}
```

Técnicas de Programación

Ejemplo creación GUI

```
public Main() {

    // Definir gestor de geometria.
    setLayout(new BorderLayout(5,5));

    Label l = new Label("Ejemplo");   // Crear etiqueta
    Button b = new Button("Pulsame 2");// Crear boton
    FotoSPane fp = new FotoSPane();    // Crear panel

    // Visualizar componentes.
    add("North",l);
    add("South",b);
    add("Center",fp);

    // Registrar escuchador.
    b.addActionListener(new Mensaje(fp,Color.yellow));
```


Ejemplo creación GUI

```
    // Repetir el proceso para el contenedor fp.

    Button b2 = new Button("pulsa");
    fp.add(b2);
    b2.addActionListener(new
    Mensaje(fp,Color.red));
}
```

Ejemplo creación GUI

```
/** Pinta el borde rojo. */
public void paint(Graphics g) {

    g.setColor(Color.red);
    g.drawRect(5,5,290,290);

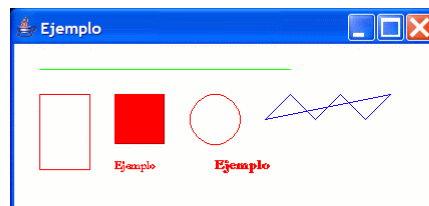
}
```


java.awt.Graphics

La clase **java.awt.Graphics**

- Proporciona métodos para dibujar figuras, fonts, imágenes, etc.
- Representa el contexto gráfico de un componente o imagen.

Ejemplo



Código del ejemplo

```
public void paint(Graphics g) {

        g.setColor(Color.green);
        g.drawLine(25,25,275,25);

        g.setColor(Color.red);
        g.drawRect(25,50,50,75);
        g.fill3DRect(100,50,50,50,true);
        g.drawOval(175,50,50,50);
```


Código del ejemplo

```
GraphicsEnvironment ge =
      GraphicsEnvironment.
                  getLocalGraphicsEnvironment();
String[] fonts =
   ge.getAvailableFontFamilyNames();

g.setFont(new Font(fonts[0],Font.PLAIN,12));
g.drawString("Ejemplo",100,125);

g.setFont(new Font(fonts[0],Font.BOLD,14));
g.drawString("Ejemplo",200,125);
```

Código del ejemplo

```
  g.setColor(Color.blue);
  int x[] = {250,275,300,325,350,375,400};
  int y[] = {  75,  50,  75,   50,  75,  50,  75};
  g.drawPolygon(x,y,6);
}
```

Técnicas de Programación

Métodos

- Líneas:
```
void drawLine(int x1, int y1, int x2, int y2);
```
- Rectángulos:
```
void drawRect(int x, int y, int width, int height);
void fillRect(int x, int y, int width, int height);
void clearRect(int x, int y, int width, int height);
void draw3dRect(int x, int y, int width, int height,
                boolean raised);
void fill3dRect(int x, int y, int width, int height,
                boolean raised);
void drawRoundRect(int x, int y, int width, int height,
                   int arcWidth, int arcHeight);
void fillRoundRect(int x, int y, int width, int height,
                   int arcWidth, int arcHeight);
```

Técnicas de Programación

Métodos (2)

- Elipses:

```
void drawOval(int x, int y, int width, int height);
void fillOval(int x, int y, int width, int height);
```

- Texto:

```
void drawString(String str, int x, int y);
void drawChars(char charArray[], int offset,
               int numChars, int x, int y);
void drawBytes(byte byteArray[], int offset,
               int numChars, int x, int y)
```

- Polígonos:

```
void drawPolygon(int[] xPoints, int[] yPoints,
                 int numPoints);
void drawPolygon(Polygon p);
```


Métodos (3)

- Imágenes:

```
boolean drawImage(Image img, int x, int y,
                  ImageObserver  observer);
boolean drawImage(Image img, int x, int y, int width, int height,
                  ImageObserver observer);
boolean drawImage(Image img, int x, int y,
                  Color bg, ImageObserver ob);
boolean drawImage(Image img, int x, int y, int width, int height,
                  Color bg, ImageObserver ob);
boolean drawImage(Image img, int dx1, int dy1, int dx2, int dy2,
                  int sx1, int sy1, int sx2, int sy2, Color bg,
                  ImageObserver ob);
boolean drawImage(Image img, int dx1, int dy1, int dx2, int dy2,
                  int sx1, int sy1, int sx2, int sy2,
                  ImageObserver ob);
```

Métodos (4)

- Color: **setColor**(Color c);

- Modos de dibujar:
  - *Modo Paint:* **setPaintMode**();
  - *Modo XOR:* **setXORMode**(Color color);

- Limitar zona de dibujo:
  - Reducir zona de pintado actual:
    void **clipRect**(int x, int y, int width, int height);
  - Establecer zona de pintado:
    void **setClip**(int x, int y, int width, int height);
  - Consultar zona actual:
    Rectangle **getClipBounds**();

La clase **Polygon**

- Constructores:
  - A partir de array de coordenadas:
    Polygon(int[] xPoints, int[] yPoints, int numPoints);
  - Creando polígono vacio y añadiendo puntos:
    Polygon();
    void **addPoint**(int x, int y);
- Métodos:
  - Rectángulo que lo abarca:
    Rectangle **getBounds**();
  - Contiene un punto:
    boolean **contains**(int x, int y);

La clase **Font**

- Constructor: **Font**(String fontName, int style, int size);
  - Ejemplo:

    Font f = new Font("TimesRoman",Font.BOLD,12);
- Constantes de estilo:

    Font.**PLAIN** Font.**BOLD** Font.**ITALIC**
  - Pueden sumarse estas constantes al construir el font.
- Métodos:

    String **getFamily**();
    String **getName**();
    int **getSize**();
    int **getStyle**();
    boolean **isBold**();
    boolean **isItalic**();
    boolean **isPlain**();

La clase **Font** (2)

- Métodos para obtener fonts instalados en el sistema:

    static Font **getFont**(String property);
    static Font **getFont**(String property, Font defaultValue);
    Formato de la propiedad: ***font-style-pointsize***
    Ejemplo: **TimesRoman-bold-16**

- La clase GraphicsEnvironment mantiene información sobre los fonts instalados.

```
GraphicsEnvironment ge =
  GraphicsEnvironment.getLocalGraphicsEnvironment();
String[] fonts = ge.getAvailableFontFamilyNames();
```

La Clase **FontMetrics**

- Métricas sobre fonts

Técnicas de Programación

La clase **Point**

- Constructores:

```
Point();
Point(int x, int y);
Point(Point p);
```

- Miembros:

```
int x;
int y;
```

- Métodos:

```
void move(int newX, int newY);
void translate(int xChange, yChange);
```

Técnicas de Programación

La clase **Dimension**

- Constructores:

```
Dimension();
Dimension(int width, int height);
Dimension(Dimension oldDimension);
```

- Miembros:

```
int width;
int height;
```


La clase **Rectangle**

- Constructores:

```
Rectangle(Point p, Dimension d);
Rectangle(int x, int y, int width, int height);
Rectangle(int width, int height);
Rectangle(Point p);
Rectangle(Dimension d);
Rectangle(Rectangle r);
Rectangle();
```

- Miembros:

```
int x;
int y;
int width;
int height;
```

La clase **Rectangle** (2)

- Métodos:

```
void move(int newX, int newY);
void translate(int xChange, yChange);
void setSize(int newWidth, int newHeight);
void grow(int widthChange, int heightChange);
void setBounds(int newX, int newY,
               int newWidth, int newHeight);
boolean contains(int x, int y);
Rectangle intersection(Rectangle anotherRect);
boolean intersects(Rectangle anotherRect);
Rectangle union(Rectangle anotherRect);
void add(Point p);
void add(int x, int y);
void add(Rectangle anotherRect);
```


La clase **Color**

- Constructores:

```
Color(int red, int green, int blue);
Color(int rgb);
Color(float red, float green, float blue);
```

- Métodos:

```
int getRed();
int getGreen();
int getBlue();
Color darker();
Color brighter();
```

La clase **Color** (2)

- Constantes:
  - Color.white
  - Color.lightGray
  - Color.gray
  - Color.darkGray
  - Color.black
  - Color.red
  - Color.pink
  - Color.orange
  - Color.yellow
  - Color.green
  - Color.magenta
  - Color.cyan
  - Color.blue

Doble Buffering

Doble Buffering

- Objetivo: Evitar parpadeo en las animaciones.
- El método **`update`** borra y repinta todo.
  - Si el tiempo en repintar es largo pueden notarse parpadeos o ver como se construye la imagen.
  - Una solución:
    - No borrar todo antes de pintar, borrar y repintar sólo los cambios.
    - Esto no siempre es aplicable, depende de lo que estemos animando.
- Doble Buffering:
  - La nueva imagen se construye en memoria y se pinta de golpe cuando está lista, sin borrar nada previamente.

Codificación doble buffering

```
Image buffer = createImage(800,600);

public void update(Graphics g) {
    // Contexto grafico del buffer
    Graphics bg = buffer.getGraphics();

    // Borrar el buffer.
    bg.setColor(getBackground());
    bg.fillRect(0,0,800,600);

    paint(bg); // Pintar en el buffer.

    g.drawImage(buffer,0,0,this); // Copiar el buffer.
}
public void paint(Graphics g) {
     // sentencias de pintar.
}
```

Componentes

Técnicas de Programación

java.awt.Component

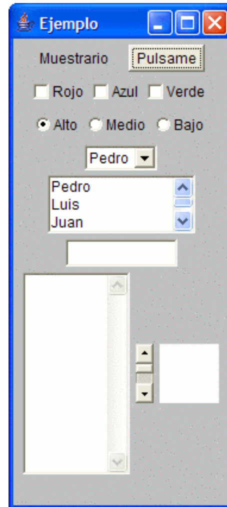
- Proporciona componentes de alto nivel para construir los interfaces de usuario.
- Clases:

Button	**Label**	**Checkbox**
Choice	**List**	**TextField**
TextArea	**Scrollbar**	**Canvas**
Menu	**MenuBar**	**PopupMenu**

- Contenedores:
  - **Panel**
  - **Frame**
  - **Dialog**
  - **ScrollPane**

Técnicas de Programación

Ejemplo: componentes



Ejemplo

Label label = **new Label**("Muestrario");

Button button = **new Button**("Pulsame");

add(label);
add(button);

Ejemplo

```
Checkbox cb1 = new Checkbox("Rojo");
Checkbox cb2 = new Checkbox("Azul");
Checkbox cb3 = new Checkbox("Verde");

add(cb1);
add(cb2);
add(cb3);
```


Ejemplo

```
CheckboxGroup cbg = new CheckboxGroup();

Checkbox r1 = new Checkbox("Alto",   cbg,true);
Checkbox r2 = new Checkbox("Medio",cbg,false);
Checkbox r3 = new Checkbox("Bajo",  cbg,false);

add(r1);
add(r2);
add(r3);
```

Ejemplo

```
Choice ch = new Choice();

ch.addItem("Pedro");
ch.addItem("Luis");
ch.addItem("Juan");
ch.addItem("Mario");

add(ch);
```


Ejemplo

```
List list = new List(3,false);

list.add("Pedro");
list.add("Luis");
list.add("Juan");
list.add("Mario");

add(list);
```

Ejemplo

```
TextField tf = new TextField(10);
add(tf);

TextArea ta = new TextArea(10,10);
add(ta);
```


Ejemplo

```
Scrollbar sb = new Scrollbar();

Canvas canvas = new Canvas();
canvas.setBackground(Color.white);
canvas.setSize(50,50);

add(sb);
add(canvas);
```

Componentes: métodos comunes

- Pintado
  - void **setForeground**(Color c);
  - void **setBackground**(Color c);
  - Color **getForeground**();
  - Color **getBackground**();
  - void **setFont**(Font f);
  - Font **getFont**();
  - FontMetrics **getFontMetrics**(Font font);
  - void **setVisible**(boolean b);
  - boolean **isVisible**();

Componentes: métodos comunes

- Pintado
  - void **repaint**();
  - void **update**(Graphics g);
  - void **paint**(Graphics g);

- El repintado no tiene porque ser inmediato. tm solicita el repintado antes de tm milisegundos.
  - void **repaint**(long tm);

- Puede solicitarse el repintado de una zona.
  - void **repaint**(int x, int y, int width, int height);
  - void **repaint**(long tm, int x, int y,int width,int height);

Componentes: métodos comunes

- Posicionamiento y Tamaño:
  - Dimension **getSize**(); void **setSize**(....);
  - void **setLocation**(Point p); Point **getLocation**();
  - void **setBounds**(Rectangle r); Rect.. **getBounds**();
  - Component **getComponentAt**(Point p);
- Indicar tamaño mínimo, máximo y preferido:
  - Dimension **getMinimumSize**();
  - Dimension **getMaximumSize**();
  - Dimension **getPreferredSize**();
    - el gestor de geometría es el que decide.
    - No hay métodos **set**: redefinir estos métodos en clases derivadas.

Componentes: métodos comunes

- Gestión de eventos:
  - La recepción de eventos puede habilitarse, deshabilitarse o consultarse:
    - void **setEnabled**(boolean b);
    - boolean **isEnabled**();

Componentes: métodos comunes

- Componentes pueden estar en dos estados según estén actualizados por el gestor de geometría o no:
  - **valid**
  - **invalid**
- Pasan al estado **invalid** si se cambia su tamaño o posición.

- Para cambiar el estado:
  - void **invalidate**();
  - void **validate**();
- validate() invoca doLayout(), que no hace nada, excepto en los contenedores, que recalcula los hijos.
  - void **doLayout**();

Componentes: métodos comunes

- Referencia al padre:

Container **getParent**();

Contenedores

Descripción

- Necesarios para organizar los componentes en grupos.
- Todo componente debe incluirse dentro de un contenedor.
- Los contenedores también son componentes.
- Tipos:
  - **Panel**: simplemente un marco.
  - **Frame**: una ventana independiente.
  - **Dialog**: ventana emergente para diálogos.
  - **ScrollPane**: panel con barras de scroll.

Contenedores: métodos comunes

- Añadir un componente al contenedor:
  - Component **add**(Component comp);
  - Component **add**(Component comp, int pos);
  - Component **add**(String name, Component comp);
- Eliminar componentes:
  - void **remove**(Component comp);
  - void **removeAll**();
- Consultar:
  - Component **getComponent**(int n)
    throws ArrayIndexOutOfBoundsException;
  - Component[] **getComponents**();
  - int **countComponents**();

Contenedor: **Panel**

- Es un marco sin nada especial.
- Constructor:

  Panel();

Contenedor: **Frame**

- Constructores:
  - **Frame**();
  - **Frame**(String frameTitle);
- Métodos:
  - void **dispose**();
  - void **setTitle**(String newTitle);
  - String **getTitle**();
  - void **setCursor**(int cursorType);
  - int **getCursorType**();
  - void **setResizable**(boolean allowResizing);
  - boolean **isResizable**();
  - void **setIconImage**(Image image);
  - Image **getIconImage**();

Frame

- Tipos de cursores disponibles:
  - **Frame.DEFAULT_CURSOR**
  - **Frame.CROSSHAIR_CURSOR**
  - **Frame.TEXT_CURSOR**
  - **Frame.WAIT_CURSOR**
  - **Frame.HAND_CURSOR**
  - **Frame.MOVE_CURSOR**
  - **Frame.N_RESIZE_CURSOR**
  - **Frame.NE_RESIZE_CURSOR**
  - **Frame.E_RESIZE_CURSOR**
  - **Frame.SE_RESIZE_CURSOR**
  - **Frame.S_RESIZE_CURSOR**
  - **Frame.SW_RESIZE_CURSOR**
  - **Frame.W_RESIZE_CURSOR**
  - **Frame.NW_RESIZE_CURSOR**

Uso de Menús en los Frames

- Los frames pueden tener barra de menús.

- Constructor: crear una barra de menús:
  - **MenuBar**();

- Añadir una barra de menús a un frame:
  - void **setMenuBar**(MenuBar mb);

Menú

- Constructor:
  - **Menu**();
  - **Menu**(String menuLabel);
  - **Menu**(String menuLabel,boolean allowTearoff);

- Añadir un menu a una barra de menús:
  - Menu **add**(Menu m);

Ejemplo

```
Frame myFrame = new Frame();

// Crear barra de menús
MenuBar myMenuBar = new MenuBar();

// Añadir la barra de menús al frame
myFrame.setMenuBar(myMenuBar);

// Crear menú
Menu fileMenu = new Menu("File");

// Añadir menú a barra de menús
myMenuBar.add(fileMenu);
```


Menú

- Las entradas del menú se pueden añadir directamente, o creando un MenuItem:
  - fileMenu.**add**("Open");
  - MenuItem saveMenuItem = **new MenuItem**("Save");
    fileMenu.**add**(saveMenuItem);
- Las entradas del menú pueden habilitarse o deshabilitarse:

  saveMenuItem.**setEnabled**(false);
- Para añadir un separador al menú usar:

  void **addSeparator**();

Menú

- Creando un submenú:

```
Menu printSubmenu = new Menu("Print");
fileMenu.add(printSubmenu);
printSubmenu.add("Print Preview");
printSubmenu.add("Print Document");
```

- Los menús pueden contener entradas de tipo Checkbox:

```
CheckboxMenuItem(String itemLabel);
boolean getState();
void setState(boolean newState);
```

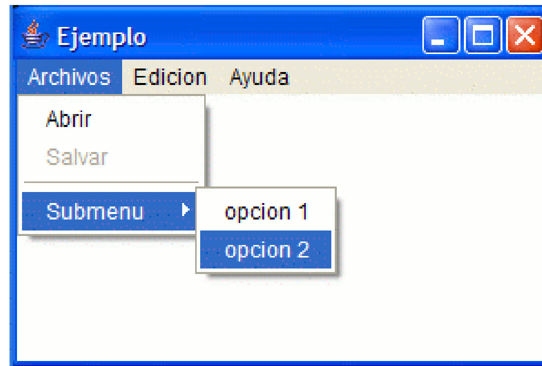

Menú

- La colocación de los menús en la barra es de izquierda a derecha.

- Puede crearse un único menú en el lado derecho (normalmente de ayuda) con:

```
void setHelpMenu(Menu helpMenu);
```

Ejemplo



Ejemplo: el menú Archivos

```
Menu ma = new Menu("Archivos");

ma.add("Abrir");

MenuItem save = new MenuItem("Salvar");
ma.add(save);
save.setEnabled(false);

ma.addSeparator();

Menu sm = new Menu("Submenu");
ma.add(sm);
sm.add("opcion 1");
sm.add("opcion 2");
```

Ejemplo: Los menús Edición y Ayuda

```
// Menu Edicion
Menu me = new Menu("Edicion");
CheckboxMenuItem cmi =
            new CheckboxMenuItem("mayus");
me.add(cmi);


// Menu Ayuda
Menu help = new Menu("Ayuda");
help.add("ayuda");
```


Ejemplo: ventana y barra de menús

```
// una ventana nueva
Frame f = new Frame();

// Barra de menus de la ventana
MenuBar mb = new MenuBar();
f.setMenuBar(mb);

// añadir menu de archivos, edicion y ayuda
mb.add(ma);
mb.add(me);
mb.setHelpMenu(help);
```

Popup menus

- Pueden asociarse menús popup con los componentes.
- Constructores:
  - **PopupMenu**();
  - **PopupMenu**(String title);
- Se añaden elementos al menú popup igual que en un menú normal.
- El menú se añade a un componente usando el método **add** del componente.

Ejemplo: Popup menús

```
PopupMenu pm = new PopupMenu("Ejemplo");

pm.add("uno");
pm.add("dos");

Canvas c = new Canvas();
c. add(pm);
```

Ejemplo: Popup menús (con)

```
// Activación de popup:
c.addMouseListener(new MouseAdapter() {

   public void mouseReleased(MouseEvent e) {
        if (e.isPopupTrigger()) {
             pm.show(e.getComponent(),
                     e.getX(),
                     e.getY());
        }
   }
});
```


Contenerdor: **Dialog**

- Son ventanas independientes con más limitaciones que un frame.
- Se crean SIEMPRE asociadas a un frame o a otro dialogo.
- Dos tipos:
  - **Modales**: Bloquean el frame asociado mientras el dialogo es visible.
  - **No modales**: no lo bloquean.

Dialog

- Contructores:
  - **Dialog**(Frame parentFrame, boolean isModal);
  - **Dialog**(Frame parentFrame, String title,
    boolean isModal);
- Métodos:
  - void **setResizable**(boolean allowResizing);
  - boolean **isResizable**();
  - void **setTitle**(String newTitle);
  - String **getTitle**();
  - boolean **isModal**()

Contenedor: **ScrollPane**

- Es un Panel con barras de desplazamiento.
- Constructores:
  - **ScrollPane**();
  - **ScrollPane**(int scrollbarOption);

- Valores válidos para el parámetro scrollbarOption:
  - ScrollPane.SCROLLBARS_ALWAYS
  - ScrollPane.SCROLLBARS_NEVER
  - ScrollPane.SCROLLBARS_AS_NEEDED // valor por defecto

ScrollPane

- Métodos:
  - void **setScrollPosition**(Point point);
  - void **setScrollPosition**(int x, int y);
  - Dimension **getViewport**(); //zona visible

- Obtener referencias a las barras de desplazamiento:
  - Adjustable **getHAdjustable**();
  - Adjustable **getVAdjustable**();

Gestores de Geometría

¿Qué son?

- Colocar componentes dentro de los contenedores.
- Estrategia de colocación basada en definir posiciones relativas para los componentes, no absolutas.
- Independiente de las dimensiones de la pantalla.
- Tipos:

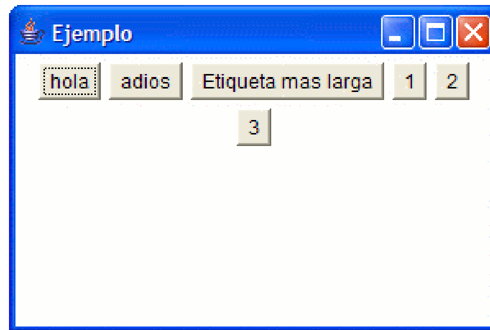
  FlowLayout GridLayout GridBagLayout
  BorderLayout CardLayout null
- Se instalan en el contenedor con:
  - void **setLayout**(LayoutManager mgr);

Gestor: **FlowLayout**

- Coloca los componentes de izquierda a derecha, línea a línea.
- Constructores:
  - **FlowLayout**();
  - **FlowLayout**(int alignment);
  - **FlowLayout**(int alignment, int hgap, int vgap);
- Valores válidos para el alineamiento:
  - FlowLayout.LEFT
  - FlowLayout.RIGHT
  - FlowLayout.CENTER

FlowLayout

- Ejemplo:



FlowLayout

- Ejemplo:

```
FlowLayout layout = new FlowLayout();
setLayout(layout);

add(new Button("hola"));
add(new Button("adios"));
add(new Button("Etiqueta mas larga"));
add(new Button("1"));
add(new Button("2"));
add(new Button("3"));
```

Gestor: **GridLayout**

- Crea una parrilla de celdas iguales que se rellena de izquierda a derecha, línea a línea.
- La parrilla la crea con el número de columnas o filas que se le indique.
  - El gestor calcula el otro parámetro.
  - El parámetro libre debe ponerse a cero.
- Constructores:
  - **GridLayout**(int rows,int cols);
  - **GridLayout**(int rows,int cols,int hgap,int vgap);

GridLayout

- Ejemplo:



GridLayout

- Ejemplo:

```
GridLayout layout = new GridLayout(2,0);
setLayout(layout);

add(new Button("hola"));
add(new Button("adios"));
add(new Button("Etiqueta mas larga"));
add(new Button("1"));
add(new Button("2"));
add(new Button("3"));
```


Gestor: **BorderLayout**

- Coloca los componentes en los cuatro puntos cardinales, o en el centro.
  - La zona centro se expande siempre para ocupar el mayor área posible.
  - Las demás zonas intentan ocupar el menor área posible.

BorderLayout

- Constructores:
  - **BorderLayout**();
  - **BorderLayout**(int hgap, int vgap);
- Métodos:
  - El método add tiene un parámetro adicional para indicar el punto cardinal.
    - Component **add**(String pos, Component comp);
    - Component **add**(Component comp, int index);
  - Constantes:
    - “North” BorderLayout.NORTH
    - “South” BorderLayout.SOUTH
    - “East” BorderLayout.EAST
    - “West” BorderLayout.WEST
    - “Center” BorderLayout.CENTER

BorderLayout

- Ejemplo:



BorderLayout

- Ejemplo:

```
BorderLayout layout = new BorderLayout();
setLayout(layout);

add(new Button("norte"),BorderLayout.NORTH);
add(new Button("este"),BorderLayout.EAST);
add(new Button("el lado oeste"),BorderLayout.WEST);
add(new Button("sur"),BorderLayout.SOUTH);
add(new Button("centro"),BorderLayout.CENTER);
```


Gestor: **GridBagLayout**

- Crea una parrilla de celdas de igual tamaño.
  - Un componente puede ocupar varias celdas.
- Constructor:
  - **GridBagLayout**();
- Hay que dar al gestor de geometría las restricciones para colocar cada componente:
  - Los conjuntos de restricciones son objetos de tipo:
    - **GridBagConstraints**
  - Para poner establecer las restricciones:
    - **setConstraints**(Component comp, GridBagConstraints gbc);

La clase **GridBagConstraints**

- **gridx**, **gridy**:
  - Posición en la parrilla donde colocar el componente.
  - Valor por defecto es GridBagConstraints.RELATIVE, que indica *"a continuación de la última posición"*.
- **gridwidth**, **gridheight**:
  - Número de celdas a ocupar por el componente.
  - Constantes:
    - GridBagConstraints.REMAINDER: ocupar hasta el final.
    - GridBagConstraints.RELATIVE: ocupar hasta el último colocado

GridBagConstraints

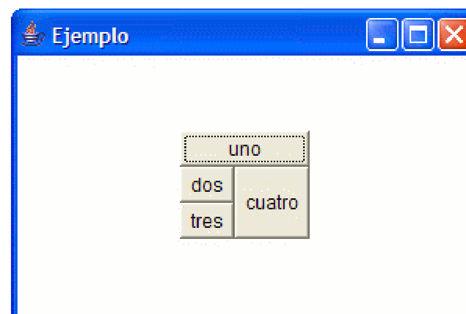
- **fill**:
  - Como rellenar el espacio si la celda es mayor que el componente.
  - Valores posibles:
    - GridBagConstraints.NONE
    - GridBagConstraints.HORIZONTAL
    - GridBagConstraints.VERTICAL
    - GridBagConstraints.BOTH
- **ipadx**, **ipady**:
  - Pixels a añadir al componente para hacerlo mayor.
- **insets**:
  - Instancia de la clase **Insets**.
  - Indica espacio a dejar alrededor del componente.

GridBagConstraints

- **anchor**:
  - Donde anclar el componente dentro de la celda.
    - Si la celda es mayor que el componente.
  - Valores posibles:
    - GridbagConstraints.CENTER (valor por defecto)
      GridbagConstraints.NORTH
      GridBagConstraints.NORTHEAST
      GridBagConstraints.EAST
      GridBagConstraints.SOUTHEAST
      GridBagConstraints.SOUTH
      GridBagConstraints.SOUTHWEST
      GridBagConstraints.WEST
      GridBagConstraints.NORTHWEST
- **weightx**, **weighty**:
  - Indica un tamaño relativo de los componentes.

GridBagConstraints

- Ejemplo:



Ejemplo: GridBagConstraints

- Ejemplo:

```
GridBagLayout layout = new GridBagLayout();
setLayout(layout);

GridBagConstraints c = new GridBagConstraints();

Button b1 = new Button("uno");
Button b2 = new Button("dos");
Button b3 = new Button("tres");
Button b4 = new Button("cuatro");
```


Ejemplo: GridBagConstraints

```
c.gridx = 0;
c.gridy = 0;
c.gridwidth = 2;
c.fill = GridBagConstraints.BOTH;
layout.setConstraints(b1,c);

add(b1);
```

Ejemplo: GridBagConstraints

```
c.gridx = 0;
c.gridy = 1;
c.gridwidth = 1;
c.fill = GridBagConstraints.BOTH;
layout.setConstraints(b2,c);

add(b2);
```


Ejemplo: GridBagConstraints

```
c.gridx = 0;
c.gridy = 2;
// el valor de c.gridwidth era 1
c.fill = GridBagConstraints.BOTH;
layout.setConstraints(b3,c);

add(b3);
```

Ejemplo: GridBagConstraints

```
c.gridx = 1;
c.gridy = 1;
// el valor de c.gridwidth era 1
c.gridheight = 2;
c.fill = GridBagConstraints.BOTH;
layout.setConstraints(b4,c);

add(b4);
```


Gestor: **CardLayout**

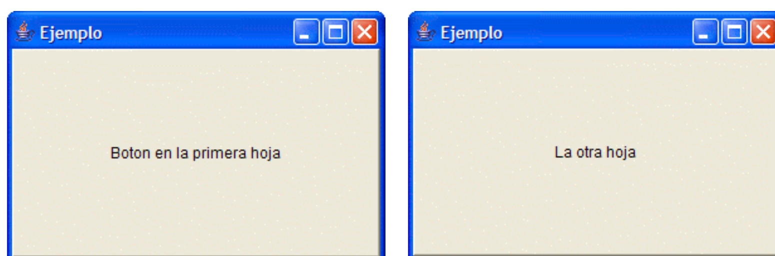
- Apila los componentes, siendo visible sólo uno de ellos.
  - Suelen apilarse varios Panel, cada uno con su propio gestor de geometría.
- Constructores:
  - **CardLayout**();
  - **CardLayout**(int hgap, int vgap);
- En el método add se asigna una etiqueta al componente gestionado.
  - Component **add**(String label, Component c);

CardLayout

- Puede seleccionarse el componente a visualizar con:
  - void **show**(Container c, String name);
  - void **first**(Container c);
  - void **last**(Container c);
  - void **next**(Container c);
  - void **previous**(Container c);

CardLayout

- Ejemplo:



Ejemplo: CardLayout

```
final CardLayout layout = new CardLayout();
setLayout(layout);
Button b1 = new Button("Boton en la primera hoja");
Button b2 = new Button("La otra hoja");
add("uno",b1);
add("dos",b2);

ActionListener al = new ActionListener() {
  public void actionPerformed(ActionEvent ev) {
     layout.next(this);
  }
};
b1.addActionListener(this);
b2.addActionListener(this);
```


Sin gestor de geometría

- Para no tener gestor de geometría puede usarse el valor null.
  setLayout(null);
- Colocar manualmente los componentes en la posición y del tamaño que se desee:
  - **setLocation**
  - **setSize**
  - **setBounds**

La clase **Insets**

- Define un área de separación entre el borde de un contenedor y sus componentes.
- Constructor:
  - **Insets**(int top, int left, int bottom, int right);
- Los contenedores implementan el método:
  - Insets **getInsets**()
    - Redefinirlo si deseamos cambiar la separación.

```
public Insets getInsets() {
        return new Insets(20, 20, 20, 20);
}
```


Eventos

Eventos

- Los eventos representan la actividad entre el sistema, los programas y los usuarios.
- Se definen varios tipos de eventos.

Funcionamiento

- Normalmente:
  - componentes generan eventos en respuesta a las acciones de los usuarios.
  - objetos del usuario escuchan y atienden los eventos generados.
- Cuando se produce un evento:
  - El generador invoca un método en todos los objetos escuchadores registrados.
  - El método que se invoca depende del tipo de evento.
- Estos métodos se definen en varias interfaces llamadas escuchadoras.
  - Las clases escuchadoras implementan interfaces escuchadoras.

Funcionamiento

- Los objetos escuchadores deben registrarse en los generadores.
  - **addXXX** y **removeXXX**
  - XXX es el nombre de la interfaz escuchadora

- Ejemplo:

```
boton.addActionListener(obj_escuchador);
boton.removeActionListener(obj_escuchador);
```


Tipos de eventos

Evento	Componente	Escuchador
ActionEvent	Button, List, MenuItem, TextField	ActionListener
ItemEvent	Checkbox, CheckboxMenuItem, Choice, List	ItemListener
FocusEvent	Component	FocusListener
KeyEvent	Component	KeyListener
MouseEvent	Canvas, Dialog, Frame, Panel, Window	MouseListener
MouseMotionEvent	Canvas, Dialog, Frame, Panel, Window	MouseMotionListener
AdjustmentEvent	Scrollbar	AdjustmentListener
WindowEvent	Dialog, Frame	WindowListener
ComponentEvent	Dialog, Frame	ComponentListener
Etc...		

Clases Adaptadoras

- Existen clases adaptadoras cuyo objetivo es evitar que las clases escuchadoras tengan que implementar todo el interfaz escuchador.
- Las clases adaptadoras implementan los métodos de las interfaces escuchadoras con un cuerpo vacío.
  - Así, las clases escuchadoras sólo tienen que extender a los adaptadores y redefinir únicamente el cuerpo de los métodos que necesitan.

Métodos

Interfaz	Métodos
ActionListener	actionPerformed(ActionEvent)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
ComponentListener ComponentAdapter	componentHidden(componentEvent) componentShown(componentEvent) componentMoved(componentEvent) componentResized(componentEvent)
FocusListener FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)
KeyListener KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
MouseListener MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)

Métodos

Interfaz	Métodos
MouseMotionListener MouseMotionAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
WindowListener WindowAdapter	windowOpened(WindowEvent) windowClosing(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent)
ItemListener	itemStateChanged(itemEvent)
TextListener	textValueChanged(TextEvent)

Eventos de Acción

- Clase de evento: **ActionEvent**.
- Debe implementarse el interfaz **ActionListener**.
  - Tiene un único método:
    - void actionPerformed(ActionEvent event);
- Método de **ActionEvent**:
    - String **getActionCommand**();
    - devuelve la etiqueta del generador
- Método de **EventObject**:
    - Object **getSource**();
    - Devuelve objeto donde se genero el evento.

Eventos de Teclado

- Clase de evento: **KeyEvent**.
- Debe implementarse el interfaz KeyListener.
  - void **keyTyped**(KeyEvent event);
  - void **keyPressed**(KeyEvent event);
  - void **keyReleased**(KeyEvent event);
- Métodos de KeyEvent:

```
// devuelve el codigo de la tecla.
int getKeyCode()

// devuelve el caracter de la tecla.
// solo para teclas normales.
char getKeyChar()

// indica si es tecla de accion:
// F1..F12, home, flechas, etc.
boolean isActionKey();
```

- La clase **InputEvent** define el manejo de los modificadores.

Códigos de Teclas

KeyEvent.F1
. . .
KeyEvent.F12
KeyEvent.LEFT
KeyEvent.RIGHT
KeyEvent.UP
KeyEvent.DOWN
KeyEvent.END
KeyEvent.HOME
KeyEvent.PGDN
KeyEvent.PGUP

KeyEvent.PRINT_SCREEN
KeyEvent.SCROLL_LOCK
KeyEvent.CAPS_LOCK
KeyEvent.NUM_LOCK
KeyEvent.PAUSE
KeyEvent.INSERT
KeyEvent.DELETE
KeyEvent.ENTER
KeyEvent.TAB
KeyEvent.BACK_SPACE
KeyEvent.ESCAPE

Eventos de Ratón

- Clase de evento: **MouseEvent.**
- Existen dos interfaces escuchadores de eventos de ratón:
  - **MouseListener**: todo menos movimiento.
    - void **mousePressed**(MouseEvent event);
    - void **mouseReleased**(MouseEvent event);
    - void **mouseClicked**(MouseEvent event);
    - void **mouseEntered**(MouseEvent event);
    - void **mouseExited**(MouseEvent event);
  - **MouseMotionListener**: eventos de movimiento.
    - void **mouseMoved**(MouseEvent event);
    - void **mouseDragged**(MouseEvent event);
- Métodos de **MouseEvent**:
    - int **getClickCount**();
    - Point **getPoint**();
    - int **getX**();
    - int **getY**();
- La clase **InputEvent** define el manejo de los modificadores.

Técnicas de Programación

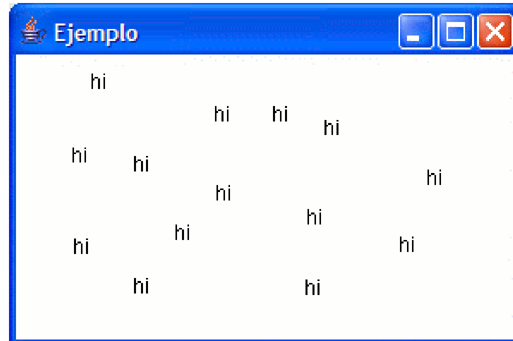
La clase **InputEvent**

- **KeyEvent** y **MouseEvent** derivan de **InputEvent**.
- Define el manejo de los modificadores de teclas y de ratón.
  - teclas de control, mayúsculas, alternativa, meta y los botones del ratón.
- Métodos de **InputEvent**:
    - **int getModifiers() // mapa de bits**
    - **boolean isShiftDown()**
    - **boolean isControlDown()**
    - **boolean isMetaDown()**
    - **long getWhen()**

Técnicas de Programación

Eventos de Ratón

- Ejemplo:



Eventos de Ratón

- Ejemplo:

```
f.addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent ev) {
      getGraphics().drawString("hi",
                                     ev.getX(),
                                     ev.getY());
    }
});
```

Manejo Mapa de bits

- Códigos:
  - InputEvent.**ALT_MASK**
  - InputEvent.**CTRL_MASK**
  - InputEvent.**META_MASK**
  - InputEvent.**SHIFT_MASK**
  - InputEvent.**BUTTON1_MASK**
  - InputEvent.**BUTTON2_MASK**
  - InputEvent.**BUTTON3_MASK**
- Ejemplo:

```
InputEvent e;
if ((e.getModifiers() & InputEvent.ALT_MASK)!= 0) {
    // la tecla ALT estaba pulsada.
}
```


Los eventos **ItemEvent**

- Clase de evento: **ItemEvent**
- Debe implementarse el interfaz **ItemListener**.
  - Tiene un único método:
    void **itemStateChanged**(ItemEvent event)
- Métodos de la clase **ItemEvent**:

```
// Checkbox, CheckboxMenuItem y Choice:
// objeto donde ocurrio el evento
ItemSelectable getItemSelectable()

// List: indice elemento seleccionado o deseleccionado.
// Checkbox, CheckboxMenuItem y Choice: etiqueta elemento.
Object getItem()

// Checkbox, CheckboxMenuItem y Choice: estado
// Estados validos: ItemEvent.SELECTED,
//                  ItemEvent.DESELECTED
int getStateChange()
```

Los eventos **AdjustmentEvent**

- Clase de evento: **AdjustmentEvent**.
- Implementar interfaz **AdjustmentListener**.
  - Tiene un único método:

    void **adjustmentValueChanged**(AdjustmentEvent e)
- Métodos de la clase **AdjustmentEvent**:
  - int getAdjustmentType()

```
// Devuelve:AdjustmentEvent.UNIT_INCREMENT
//          AdjustmentEvent.UNIT_DECREMENT
//          AdjustmentEvent.BLOCK_INCREMENT
//          AdjustmentEvent.BLOCK_DECREMENT
//          AdjustmentEvent.Track
```